Conflict resolution for pipelined layered LDPC decoders

Cédric Marchand, Jean-Baptiste Doré, Laura Conde Canencia, Emmanuel Boutillon

CONFLICT RESOLUTION FOR PIPELINED LAYERED LDPC DECODERS

Cédric Marchand[*†], *Jean-Baptiste Doré*[*], *Laura Conde-Canencia*[†], *Emmanuel Boutillon*[†]

[*]NXP Semiconductors, Campus Effiscience, Colombelles BP20000 14906 Caen cedex 9, France.
[†]Université Européenne de Bretagne, Lab-STICC, CNRS, UBS, BP 92116 56321 Lorient, France.
Email: cedric.marchand@univ-ubs.fr

ABSTRACT

Many of the current LDPC implementations of DVB-S2, T2 or WiMAX standard use the so-called layered architecture combined with pipeline. However, the pipeline process may introduce memory access conflicts. The resolution of these conflicts requires careful scheduling combined with dedicated hardware and/or idle cycle insertion. In this paper, based on the DVB-T2 example, we explain explicitly how the scheduling can solve most of the pipeline conflicts. The two contributions of the paper are 1) how to split the matrix to relax the pipeline conflicts at a cost of a reduced maximum available parallelism 2) how to project the problem of the research of an efficient scheduling to the well-known "Travelling Salesman Problem" and use a genetic algorithm to solve it.

***Index Terms*—Low-density parity-check code (LDPC), memory conflicts, scheduling, genetic algorithm, layered decoding.**

I. INTRODUCTION

Low Density Parity-Check (LDPC) codes [1] have gained a lot of attention due to their remarkable error correcting capabilities. Among all the published work on LDPC, the approach introduced in [2] led to the concept of structured codes which are now included in standards such as DVB-S2 and DVB-T2 [3] for digital video broadcasting, Wireless Local Area Networks (WiFi) (IEEE 802.11n), Wireless Metropolitan Area Networks (WiMAX) (802.16e) [4] and Wireless Regional Area Networks (WRAM) (IEEE 802.22). These structured codes or architecture-aware codes (AA-LDPC [5]) can be efficiently implemented using a semi-parallel architecture [6], [7], [8], a block-serial architecture [9], [10], [11], or a layered decoder architecture [12], [13], [14].

The turbo-decoding message-passing algorithm, introduced by Mansour [5], [15] and then referred to as layered decoding by Hocevar [14], presents the following main advantages over the standard decoding algorithm: 1) faster convergence speed by a factor of two in terms of decoding iterations and 2) reduced decoder complexity. Moreover, the use of a Soft-Output (SO) based Check Node Processor (CNP)[9], [12], [13], [14], [16] significantly reduces memory requirements.

The throughput of the layered decoding architecture can be easily doubled by pipelining, but problems of memory conflicts arise. In [13] and [16] the authors present a solution based on the computation of the variation (or delta) of the SO metrics to allow concurrent updates. The computation of this SO update needs either a costly memory access or an increase in the clock frequency by a factor of two. In [6], the use of idle time is proposed to deal with the conflicts. However this solution decreases the throughput. The scheduling of the SO in [10] reduced the use of idle time by using CNP able to deliver its outputs values in a different order than its input values, which increase the complexity of the CNP architecture. Finally, we should also note that [9], [10] propose an appropriate scheduling of the check node as a solution to avoid pipeline conflict but in none of them, the idea is fully developed.

In this paper, we focus on the conflicts due to the pipelining of a layered decoder and we propose solutions with an example on the DVB-T2 matrices. We first explain the reordering of the matrices depending on the required parallelism. The new reordered matrices solve by itself a part of the pipeline conflicts. In order to find an efficient scheduling to solve the remaining conflict, we show that the research of an efficient scheduling is equivalent to the well known "Travelling Salesman Problem". Thus, all the numerous methods described in the literature to solve the former problem can be used for our scheduling problem. In this paper, we present scheduling results using a genetic algorithm.

The remainder of the paper is organized as follows: Section II presents the layered belief propagation schedule and the memory conflicts. Section III shows how the spliting process can reduce the number of conflicts. Section IV explains how most of the remaining conflicts can be avoided by an appropriate scheduling of the layers. A method based on genetic algorithm is then proposed to find efficient scheduling for all the DVB-T2 code rates and frame types.

II. LAYERED BELIEF PROPAGATION SCHEDULING

The AA-LDPC codes are constructed from a $m \times n$ base matrix H_{base}. All positive elements of H_{base} have values

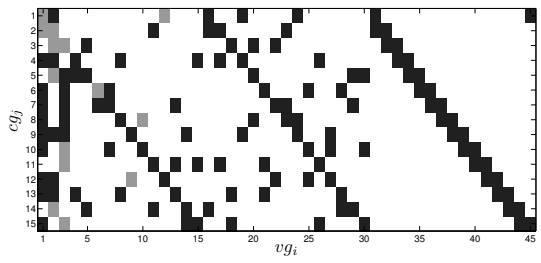


Fig. 1. Base DVB-T2 Matrix

given by $shift \in \{0, ..., P\}$ and are expanded as $P \times P$ shifted identity matrix. The elements of H_{base} with '-1' values are expanded as a $P \times P$ zero matrix. Fig. 1 shows a graphical representation of the $(m, n) = (15, 45)$ H_{base} matrix of the rate-$^2/_3$ short-frame DVB-T2 LDPC parity check matrix, with $P = 360$. cg_j stands for the j^{th} group of $P = 360$ Check Nodes (CN) and vg_i for the i^{th} group of $P = 360$ Variable Nodes (VN). The size of the expanded LDPC matrix is then $(15 \times 360, 45 \times 360) = (5400, 16200)$. The black squares represent shifted identity matrices while the grey squares are double identity sub-matrices which are the cause for another type of conflict solved in [17]. They are beyond the scope of our work. Note the DVB-T2 structured matrices are efficient for highly parallel decoding using the layered decoder algorithm.

II-A. Layered decoder algoritm

In the horizontal layered decoding algorithm, a VN is represented by a Soft Output (SO_v) value. The SO_v value is first initialized by the Channel Log Likelihood Ratio ($LLR = log(P(v = 0)/P(v = 1))$). Then the decoding proceeds iteratively until all the parity-checks of the code are verified or a maximum number of iterations is reached. For layered decoding, one iteration is split into sub-iterations and each sub-iteration processes one layer. A layer can be made of one or several CNs and the sub-iteration consists in updating all the VNs connected to the CNs of the layer.

The update of the VNs connected to one CN is done serially in three steps. First, the messages from VN to CN ($M_{v\to c}$) are calculated using equation (1) with $M_{c\to v} = 0$ during the first iteration.

$$M_{v\to c} = SO_v - M_{c\to v} \tag{1}$$

The second step is the serial $M_{c\to v}$ update, where $M_{c\to v}$ is a message from CN to VN. For implementation convenience, the sign (2) and the absolute value (3) of the messages are updated separately. Let v_c be the set of all the VN connected to the CN c and v_c/v be v_c without v.

$$sign(M^{new}_{c\to v}) = \prod_{v'\in v_c/v} sign(M_{v'\to c}) \tag{2}$$

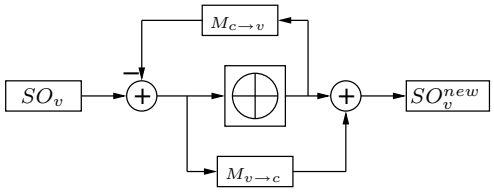


Fig. 2. SO based CNP

$$|M^{new}_{c\to v}| = f\left(\sum_{v'\in v_c/v} f(|M_{v'\to c}|)\right) \tag{3}$$

where $f(x) = \ln\tanh\left(\frac{x}{2}\right)$. Equation (3) can be implemented using a sub-optimal algorithm such as min-sum algorithm [18], normalized min-sum algorithm or the λ -min algorithm [19]. The third step is the calculation of the SO_{new} value using (4) .

$$SO^{new}_v = M_{v\to c} + M^{new}_{c\to v} \tag{4}$$

From these equations, the CNP architecture in Fig. 2 can be derived. The left adder of the architecture performs equation (1) and the right adder performs equation (4). The central part is in charge of the serial $M_{c\to v}$ update.

Several CNs may be grouped together to form a layer, whenever the column weights in the layer does not exceed one. In other words, a given VN is connected at most to a single CN of a layer. The layered decoder architecture is mainly based on P CNPs that read serially the VGs linked to the CG and then the P CNPs write back the result to the VGs in the same order.

II-B. Non pipelined CNP

The chronogram in Fig. 3 illustrates a non-pipelined CNP. The CNP first reads the SO. Then after a given number of clock cycles ϵ, i.e. the CN latency, the CNP writes back the result of the calculation. We can see on this chronogram that the CNP starts to read the new set of variable nodes $v_{c_{i+1}}$ only when all the previous v_{c_i} have been calculated. The corresponding throughput is given by:

$$D_1 = \frac{K.F_{clk}}{(2d_c + \epsilon).\frac{M}{P}.N_{it}} \quad bit.s^{-1} \tag{5}$$

where N_{it} is the number of iterations to decode a codeword, M is the number of CN, P is the number of CNPs working in parallel, d_c is the average number of VNs linked to a CN, F_{clk} is the clock frequency and K is the number of information bits in a codeword.

II-C. Pipelining

Pipelining allows a more efficient use of the CNP and an increase of the throughput [6] [11] [10]. The pipelining consists in reading the v_{c_i} of one sub-iteration while writing

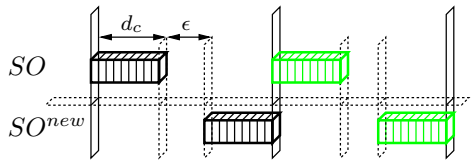


Fig. 3. Chronogram of a non pipelined CNP

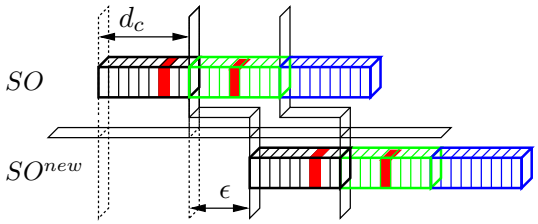


Fig. 4. Chronogram of a pipelined CNP without idle time

on $v_{c_{i-1}}$ the result of the previous sub-iteration. This means that as soon as the reading of one sub-iteration is finished, a new one is started. The chronogram is given in Fig. 4 and the corresponding throughput is given by the following equation:

$$D_2 = \frac{K.F_{Clk}}{d_c.\frac{M}{P}.N_{it} + d_c + \epsilon} \quad bit.s^{-1} \tag{6}$$

The pipelined architecture offers at least two times greater throughput compared to the non pipelined one: $\frac{D_2}{D_1} \approx 2 + \frac{\epsilon}{d_c}$. We will see in the next subsection that this architecture can lead to memory conflicts.

II-D. The problem of memory conflicts

In this section, we show that two type of memory conflicts can occur.

"Type $i+1$" memory conflict: In Fig. 4 a common variable SO_{com} (filled square) is used in two successive sub-iterations. During the second sub-iteration, the SO_{com} is still not updated from the previous sub-iteration and the result of the current sub-iteration will overwrite the result of the previous sub-iteration. This is known as a 'cutting edge' problem because it is equivalent to a cut in an edge of the Tanner graph representation of the matrix. Let vg_{cg_i} be the set of all the vg_i connected to cg_j. During pipelining, the P CNPs write on vg_{cg_i} while the P CNPs read on vg_{cg_j}. The layers cg_i and cg_j can work one after the other without memory access conflict when the two groups don't share any common variable [1]. Mathematically speaking, this constraint can be expressed by:

$$vg_{cg_i} \cap vg_{cg_j} = \emptyset$$

[1]As mentionned in [10], if $dc > \epsilon$, it is still possible to avoid memory conflict between two groups sharing the same common variable SO_{com} by an appropriate scheduling of the SO_{com} inside the two consecutive layers.

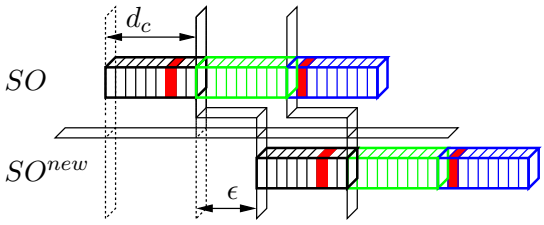


Fig. 5. Conflict due to pipelining at $i+2$

For example, we can check in Fig. 1 that there is no vg_i in common between the set of vg_i linked to the group of check node number one cg_1 and eleven cg_{11}:

$$\{vg_{cg_1}\} \cap \{vg_{cg_{11}}\} =$$

$$\{1,2,12,16,19,24,31,45\} \cap \{3,11,13,15,17,21,26,40,41\} = \emptyset$$

Thus the decoding of cg_1 and cg_{11} can be processed consecutively without memory conflict. We have to take care that the next sub-iteration does not use the same SO as the previous one.

"Type $i+2$" memory conflict: Fig. 5 illustrates that the same consideration must be taken with the $i+2$ sub-iteration because of the latency ϵ. In this figure, one value of SO generated by layer i is writen in memory after it is read by layer $i+2$. This situation also leads to a memory conflict and can be avoided by appropriate scheduling of the layers.

III. CONFLICT REDUCTION BY GROUP SPLITTING

To achieve the minimum required throughput of 90Mbps in the DVB-T2 standard, parallel processing of only a fraction of the 360 CNs is enough [8] [9]. In [9], the authors have used 45 CNPs, therefore splitting the group of 360 CNs is considered. In [8] and [9], the splitting process has already been done implicitly through memory mapping. In the next subsection, we will show how to reorder the structured matrices initially designed for a parallelism of 360 to matrices designed for a parallelism of $360/S$, where S is the number of splits.

III-A. Construction of the sub-matrices

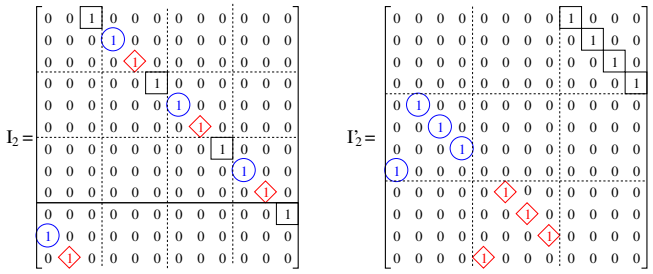
Let us define P_s the number of CN working in parallel after a split. P, P_s and S are then linked by the equation:

$$S \times P_s = P$$

The construction process of the new matrix relies on the permutation of the rows and the columns defined as:

$$\sigma(i) = (i \mod S)P_s + \lfloor i/S \rfloor \tag{7}$$

where $\lfloor x \rfloor$ is the largest integer not greater than x. We first permute the row using (7), where i is the row number. Then we permute the columns in the same way.



(a) Before split (b) After split

Fig. 6. Shifted identity matrix

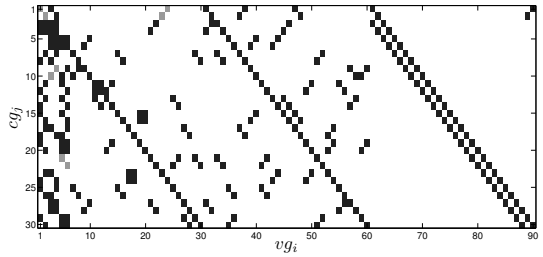


Fig. 7. New base DVB-T2 Matrix after factor 2 split

III-B. Example

Let us consider the following example of a shifted identity matrix $\mathbf{I_2}$ of size $P = 12$ in Fig. 6(a). After reordering the rows and columns using equation (7) with $S = 3$ and $P_s = 4$, we obtain the new matrix $\mathbf{I'_2}$ in Fig. 6(b). Note that $\mathbf{I'_2}$ is a structured matrix made of identity matrices of size P_s.

III-C. Results

Fig. 7 illustrates the new base matrix after a split by a factor of 2. We can see that the new base matrix is sparser than the base matrix in Fig. 1 (in terms of identity matrix density). In fact, the number of identity matrices increases by 2 while the size of the base matrix is increased by 4. Increasing the split decreases the risk of memory conflicts due to pipelining. An appropriate scheduling of the groups of CN can solve the remaining conflicts.

IV. CONFLICT RESOLUTION BY SCHEDULING

A schedule provides timing information about a series of arranged events. To avoid the cutting edge conflict, we explore the scheduling of the layers. After defining the scheduling strategy, we show that this problem is an instance of the well known "Traveling Saleman Problem". This problem can be efficiently solved by a Genetic Algorithm (GA).

IV-A. Scheduling of the layers

The schedule is done in the set of groups of CNs $cg = \{cg_1, cg_2, ..., cg_{mg}\}$. We define a schedule sequence index π, where π is one permutation in the set $\{1, 2, ..., mg\}$. The number of conflicts due to the pipelining between two check node groups , cg_i and cg_j is given by equation (8).

$$c(i,j) = |vg_{cg_i} \cap vg_{cg_j}| \tag{8}$$

The number of conflicts after one full iteration using scheduling π is given by equation (9).

$$c_{it}(\pi) = \left(\sum_{i=1}^{i=mg-1} c(\pi(i), \pi(i+1)) \right) + c(\pi(mg), \pi(1)) \tag{9}$$

where the second term is the cost between the last layer of an iteration and the first layer of the following iteration. We have to find the optimal permutation π_{opt} that gives the smallest number of conflicts; this can be translated into an optimization problem which consists in the minimization of the cost function:

$$\pi_{opt} = \arg\min\{c_{it}(\pi), \pi \in \Pi\} \tag{10}$$

where Π is the group of all the possible permutations of π. The use of graph theory is dedicated to solve this kind of problem.

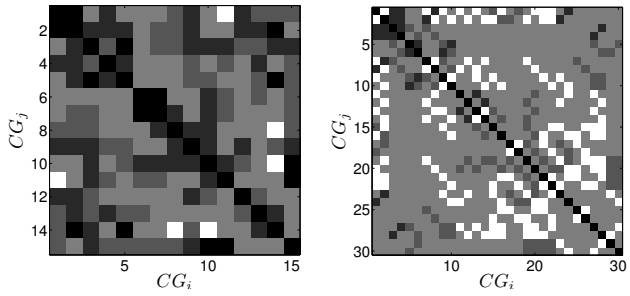
IV-B. The Traveling Salesman Problem (TPS)

Finding the schedule to avoid cutting edge is described by the minimization problem (10). An equivalent formulation in terms of graph theory is: given a complete weighted graph (where the node would represent a group of CNs, and the number of cutting edge would be the cost of the edge), find a Hamiltonian cycle with the least cost. This problem is also known as a TSP [20]. The TSP statement is as follows: given a number of cities and the cost of travelling from one city to any other city, what is the least-cost round-trip route that visits each city exactly once and then returns to the starting city. In our case a town is a group of check node and the travelling cost is the number of cutting edges (8).

The first step is to build the cost matrix $H_c = \{H_c(i,j) = c(i,j), i,j \in [1, mg]^2\}$. This matrix gives the number of cutting edges for each possible couple of groups of CN cg_i and cg_j. The cost matrix for a rate 2/3 short frame is illustrated in Fig. 8(a). On this graphical representation, a white square is for no cutting edge and a grey square means one (light grey) or more (darker grey) cutting edge.

We can see in Fig. 8(a) that there are only three couples $cg_{1,11}$, $cg_{8,14}$ and $cg_{10,14}$ that can perform consecutively without conflicts. Fig. 8(b) shows the cost matrix after a split factor of 2. The new matrix offers more possible couples without memory conflict (from 3% to 20% after splitting by two). The split process gives fewer cutting edges and a greater degree of freedom for scheduling.

The problem of trying all permutations ($|Pi| = mg!$) and selecting the minimum cost (10) is NP hard in $O(mg!)$. For a long frame of rate $^1/_4$ and split 4, the number of cities is 540. Thus a suboptimal or heuristic algorithm is needed to solve the problem.



(a) Without split (b) With a split factor of 2

Fig. 8. Cost matrix

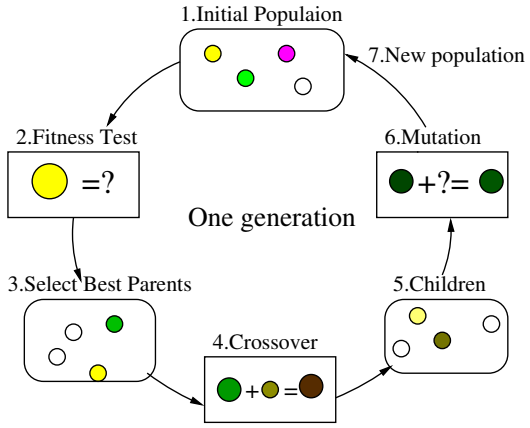


Fig. 9. Genetic algorithm

IV-C. Principle of Genetic Algorithm

Genetic algorithms [21], [20] use techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover. The genetic algorithm process is summarized in Fig. 9. First, many solutions are randomly generated to form an initial population. Then individual solutions are selected through a two round tournament selection using the fitness function given by equation (9). The next step is to generate the 'children' through a two point crossover between solutions previously selected. The last step is the mutation of the children for diversity purpose by swapping two randomly chosen cities. This generational process is repeated until a null cost is found for an individual solution or a fixed number of generations is reached.

The result of the genetic algorithm applied to the scheduling problem is presented in the next section.

IV-D. Results

In this section, we present the result of the genetic algorithm. The maximum number of generation and the initial population were set to 1000. Every solution was obtained in less than 15 minutes on a standard desktop processing unit.

		Code rate						
S	P_s	1/4	1/2	3/5	2/3	3/4	4/5	5/6
1	360	1	1	0	0	0	0	0
2	180	2	2	1	0	1	0	0
3	120	2	2	1	1	1	1	0
4	90	2	2	1	1	1	2	1
5	72	2	2	1	1	2	2	1
6	60	2	2	2	2	2	2	1
8	45	2	2	2	2	2	2	1
9	40	2	2	2	2	2	2	2

Table I. Scheduling solutions for short frames

		Code rate					
S	P_s	1/2	3/5	2/3	3/4	4/5	5/6
1	360	2	1	1	1	0	0
2	180	2	1	2	1	1	0
3	120	2	1	2	2	1	1
4	90	2	1	2	2	1	1
5	72	2	1	2	2	1	1
6	60	2	2	2	2	2	1
8	45	2	2	2	2	2	2
9	40	2	2	2	2	2	2

Table II. Scheduling solutions for long frames

In order to take into account the "type $i+2$" cutting edge defined in section II.D, i.e $vg_{cg_i} \cap vg_{cg_{i+2}} = \emptyset$, we modify the cost function c_{it} defined in (9) as:

$$c'_{it}(\pi) = \alpha c_{it}(\pi) + \sum_{i=1}^{mg} c(\pi(i), \pi((i+2) \bmod mg)) \quad (11)$$

where $\alpha \in N^+$ and is high enough to give an absolute priority to the $i+1$ conflicts against the $i+2$ conflicts. The next subsection gives some results using a genetic algorithm to find an efficient schedule.

Table I and Table II present the results found for the DVB-T2 LDPC decoder. In these tables, a '0' means that the genetic algorithm found no solution that avoids "type $i+1$" conflict. A '1' (respectively '2') means that it found a scheduling solution without type $i+1$ conflict (respectively without type $i+1$ and type $i+2$ conflicts). We can check that for $P_s = 40$, there are schedules without conflicts at $i+2$ for all code rates and frame types. Note that, after a scheduling at $i+1$, the remaining conflicts due to the latency ϵ can be avoided using a scheduling of the vg_i inside the layers [10]. This option allows a parallelism of up to 120 for long frames and 90 for short frames.

V. CONCLUSION

Pipelining a layered decoder doubles the throughput but leads to memory conflicts. The proposed reordering of the matrices reduces the parallelism and creates a sparser base matrices. Using the new base matrices, schedules without conflicts can be found. Due to the huge amount of possible solutions after reordering, a genetic algorithm is used to find the best schedule. This algorithm finds schedules that avoid

conflicts with the next sub-iteration $(i+1)$ and with the second next sub-iteration $(i+2)$. Although this article explains the process for matrices defined by the DVB-T2 standard, the same process can be used for structured matrices such as the ones defined by the WiMAX standard. Future work is focused on hardware implementation and evaluation of the performance in terms of area and throughput.

VI. REFERENCES

[1] R. Gallager, *Low-Density Parity-Check Codes*. PhD thesis, Cambridge, 1963.

[2] E. Boutillon, J. Castura, and F. Kschischang, "Decoder first code design," in *In Proc. 2nd International Symposium on Turbo Codes & Related Topics*, (Brest, France), pp. 459–462, Sept. 2000.

[3] "Frame stucture channel coding and modulation for the second generation digital terrestrial television broadcasting system (DVB-T2)," *DVB Document A122*, 2008.

[4] "Air interface for fixed and mobile broadband wireless access systems," in *P802.16e/D12 Draft*, (Washington, DC, USA), pp. 100–105, IEEE, 2005.

[5] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration VLSI Systems*, vol. 11, pp. 976–996, Dec. 2003.

[6] Y. Sun, M. Karkooti, and J. Cavallaro, "High throughput, parallel, scalable LDPC encoder/decoder architecture for OFDM systems," in *Design, Applications, Integration and Software, 2006 IEEE Dallas/CAS Workshop on*, (Richarson, USA), pp. 39–42, Oct. 2006.

[7] F. Kienle, T. Brack, and N. Wehn, "A synthesizable IP core for DVB-S2 LDPC code decoding," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, (Munich, Germany), pp. 100–105, IEEE Computer Society, Mar. 2005.

[8] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, and M. Falcao, "Flexible parallel architecture for DVB-S2 LDPC decoders," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, (Washington, USA), pp. 3265–3269, Nov. 2007.

[9] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 2, (Munich, Germany), pp. 1–6, Mar. 2006.

[10] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A minimum-latency block-serial architecture of a decoder for IEEE 802.11n LDPC codes," in *Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on*, (Atlanta, USA), pp. 236–241, Oct. 2007.

[11] T. Bhatt, V. Sundaramurthy, V. Stolpman, and D. McCain, "Pipelined block-serial decoder architecture for structured LDPC codes," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 4, (Toulouse, France), p. IV, May 2006.

[12] T. Brack, M. Alles, F. Kienle, and N. Wehn, "A synthesizable IP core for WIMAX 802.16e LDPC code decoding," in *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, (Helsinki, Finland), pp. 1–5, Sept. 2006.

[13] M. Rovini, F. Rossi, P. Ciao, N. L'Insalata, and L. Fanucci, "Layered decoding of non-layered LDPC codes," in *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, (Dubrovnick, Croatia), pp. 537–544, Sept. 2006.

[14] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, (Austin, USA), pp. 107–112, Oct. 2004.

[15] M. Mansour and N. Shanbhag, "Low-power VLSI decoder architectures for LDPC codes," in *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, (Monterey, USA), pp. 284–289, Aug. 2002.

[16] E. Boutillon and F. Guilloud, "LDPC decoder, corresponding method, system and computer program," *US patent 7,174,495 B2*, Feb. 2007.

[17] C. Marchand, J.-B. Doré, L. Conde-Canencia, and E. Boutillon, "Conflict resolution by matrix reordering for DVB-T2 LDPC decoders," in *Global Telecommunications Conference, 2009. GLOBECOM '09. IEEE "accepted"*, (Honolulu, USA), Oct. 2009.

[18] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on communications*, vol. 47, pp. 673–680, May 1999.

[19] F. Guilloud, E. Boutillon, and J.-L. Danger, "lambda-min decoding algorithm of regular and irregular LDPC codes," *Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics*, Sept. 2003.

[20] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, 1999.

[21] J. H. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992.